

2

AD-A251 927



# Performance of Redundant Disk Array Organizations in Transaction Processing Environments\*

Antoine N. Mourad<sup>†</sup>  
W. Kent Fuchs  
Daniel G. Saab

DTIC  
ELECTE  
JUN 24 1992  
S A D

Center for Reliable and High-Performance Computing  
Coordinated Science Laboratory  
University of Illinois  
1101 W. Springfield Ave  
Urbana, Illinois 61801

June 11, 1992

This document has been approved  
for public release and sale; its  
distribution is unlimited.

## Abstract

In this paper, we study the performance of two redundant disk array organizations in a transaction processing environments and compare it to that of mirrored disk organizations. Redundant disk arrays and mirrored disks are used for providing rapid recovery from media failures in systems requiring high availability. We examine three different organizations: mirrored disks, data striping with rotated parity (RAID5) and parity striping. Mirrored disks incur a 100% storage overhead. The other two organizations are much less costly in terms of storage requirements (10% storage overhead for a 10 disk array) but they provide lower throughput than mirrored disks. RAID5 provides high data transfer rates by striping the data over multiple disks. It also provides better load balancing over the disks in the array. At the same time, data striping increases disk arm use which can lead to longer queuing delays. In OLTP environments, because

\*This research was supported in part by the National Aeronautics and Space Administration (NASA) under Contract NAG 1-613 and in part by the Department of the Navy and managed by the Office of the Chief of Naval Research under Grant N00014-91-J-1283.

<sup>†</sup>Ph. (217) 244-7180, Fax (217) 244-5686, email: mourad@crhc.uiuc.edu



of the nature of I/O requests namely a large number of small size requests, disk arms are a more valuable resource than data transfer bandwidth. Hence, parity striping was proposed as an alternative to data striping. It provides rapid recovery from failure at the same low storage cost without interleaving the data over multiple disks. In this study, we use data from real applications to evaluate and compare the performance of the above three organizations.

**Keywords:** Disk Arrays, Transaction Processing, Media Recovery, Performance Evaluation, Trace-Driven Simulation, Database Systems, Parallel I/O.

## 1 Introduction

Reliable storage is a necessary feature in on-line transaction processing (OLTP) systems requiring high availability. Media failure in such systems is traditionally dealt with by periodically generating archive copies of the database and by logging updates to the database performed by committed transactions between archive copies into a redo log file. When a media failure occurs, the database is reconstructed from the last copy and the log file is used to apply all updates performed by transactions that committed after the last copy was generated. In such a case, a media failure causes significant down time and the overhead for recovery is quite high. For large systems, e.g., with over 50 disks, the mean time to failure (MTTF) of the permanent storage subsystem can be less than 25 days\*. Mirrored disks have been employed to provide rapid media recovery [1]. However, disk mirroring incurs a 100% storage overhead which is prohibitive in many cases. Redundant Disk Array (RDA) organizations [3, 8] provide an alternative for maintaining reliable storage. However, even when disk mirroring or RDAs are used, archiving and redo logging may still be necessary to

---

\* Assuming an MTTF of 30,000 hours for each disk.

protect the database against operator errors or system software design errors.

Chen *et al.* [2] compared the performance of RAID0, RAID1, and RAID5 systems<sup>†</sup>. They used a synthetic trace made of a distribution of small requests representing transaction processing workloads combined with a distribution of large requests representing scientific workloads. They used actual disk measurements on an Amdahl 5890. Gray *et al.* [3] proposed the parity striping organization and used some simple analytical models to derive the minimum (zero load) response time and the throughput at 50% utilization for fixed size requests. Their results suggest that parity striping is more appropriate than RAID5 for database and transaction processing systems. Menon and Mattson [7] analyzed the performance of RAID5 systems in the transaction processing environment using analytical models. They compared the performance of arrays made of different size building blocks and studied the effect of caching.

The following section introduces the two redundant disk array organizations discussed in this paper. Section 3 describes the traces used in our simulations. In Section 4, we present the experiments conducted and discuss the results. Finally, Section 5 presents some conclusions.

---

<sup>†</sup>Data striping without redundancy.

Statement A per telecom  
Dr. Clifford Lau  
ONR/Code 1114  
Arlington, VA 22217-5000 NWW 6/23/92

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability	
Date	Special
A-1	

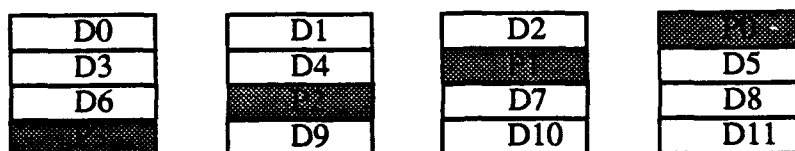


Figure 1: RAID with rotated parity on four disks.

## 2 Redundant Disk Arrays

### 2.1 Data Striping

Striped disk arrays have been proposed and implemented for increasing the transfer bandwidth in high performance I/O subsystems [5, 6, 9, 10]. In order to allow the use of a large number of disks in such arrays without compromising the reliability of the I/O subsystem, redundancy is sometimes included in the form of parity information. Patterson *et al.* [8] have presented several possible organizations for Redundant Arrays of Inexpensive Disks (RAID). One interesting organization is RAID with rotated parity in which blocks of data are interleaved across  $N$  disks while the parity of the  $N$  blocks is written on the  $N + 1^{st}$  disk. The parity is rotated over the set of disks in order to avoid contention on the parity disk. Figure 1 shows the array organization with four disks. The organization allows both large (full stripe) concurrent accesses or small (individual disk) accesses. For a small write access, the data block is read from the relevant disk and modified. To compute the new parity, the old parity has to be read, XORed with the new data and XORed with the old data. Then the new data and new parity can be written back to the corresponding disks. Stonebraker *et al.* [11] have advocated the use of a RAID organization to provide high availability in database systems.

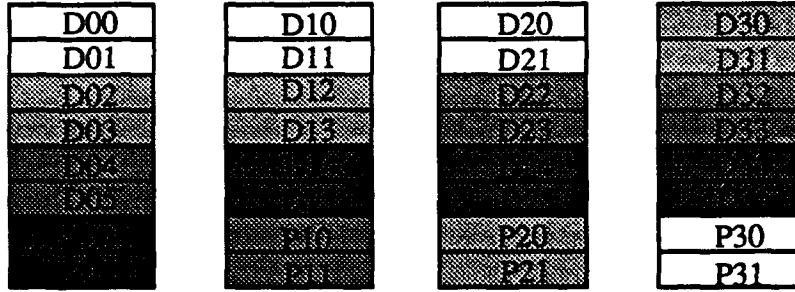


Figure 2: Parity striping of disk arrays.

## 2.2 Parity Striping

Gray *et al.* [3] studied ways of using an architecture such as RAID in OLTP systems. They argued that because of the nature of I/O requests in OLTP systems, namely a large number of small accesses, it is not convenient to have several disks servicing the same request. Hence, the organization shown in Figure 2 was proposed. It is referred to as parity striping. It consists of reserving an area for parity on each disk and writing data sequentially on each disk without interleaving. For a group of  $N + 1$  disks, each disk is divided into  $N + 1$  areas one of these areas on each disk is reserved for parity and the other areas contain data.  $N$  data areas from  $N$  different disks are grouped together in a *parity group* and their parity is written on the parity area of the  $N + 1^{st}$  disk.

## 3 Workload and System Model

To evaluate the different redundant disk array organizations, we have used data from operational OLTP systems, namely Tandem NonStop systems. The data were extracted from log files generated by the Transaction Monitoring Facility (TMF) [12] during normal system

Table 1: Disk parameters.

Max. latency	16.6 ms
Max. seek	47 ms
tracks per platter	1260
sectors per track	52
bytes per sector	512
number of platters	15

operation. The log entries contained transaction status information, before and after images of modified data, names of accessed files and disks as well as timing information. Using the log entries, we construct a trace of update accesses performed by each transaction before it commits or aborts as well as information on periodic checkpoints.

Using this data, we simulate the behavior of the database buffer and the I/O subsystem. We assume that the system is I/O bound and hence we ignore cpu processing times. The disk parameters used in the simulations are shown in Table 1. For seek time, a square root function is used for seek distances less than 20% of the number of cylinders. Beyond that point, a linear function is used.

We use three different traces from three different applications. The first trace was collected at a company issuing credit cards to individuals. The application traced is the end of month processing of customer accounts. The second trace was generated during the operational test of the Stock Point Logistics Intergrated Communications Environment (SPLICE). SPLICE is used by the US Navy Supply Systems Commands to manage stocks and relay orders to supply centers. The third trace was collected from a test application used in TMF software development. Table 2 shows the characteristics of the traces used.

Table 2: Trace characteristics.

	Trace 1	Trace 2	Trace 3
# of accesses	356730	200549	22457
duration	2hr 22min	57min	12min
# of transactions	14294	16832	8795
# of checkpoints	208	30	14
# of data disks	5	5	5

When a reference to a block that is not in the buffer occurs, the block is read into the database buffer while another block is chosen for replacement. If the replaced block is dirty, it has to be written to disk. The buffer replacement policy is LRU. Dirty blocks are written to disk either by the buffer replacement algorithm or during periodic checkpoints. These checkpoints are generated in order to reduce the amount of redo actions necessary to recover from a system crash [4]. Information on the placement of checkpoints is extracted from the log files. During periodic checkpoints, multiblock requests are issued to write back consecutive blocks while the I/O requests initiated by buffer replacement are single block I/O's.

The log records do not provide any information on read accesses. We only use the update accesses extracted from the log file to compare the different disk array organizations. There is still some read I/O performed since when a block is referenced for updating and it is not in the buffer it is read into the buffer and then updated. It is important to note that the impact of the write requests on the performance is more important than that of read requests because of the need to write both mirrors or to update both the data and parity blocks whenever a write request is issued.

I/O to the log file is not included in this evaluation. The reason for this is that log files are

usually placed on separate disks. In addition, the I/O to the log is always sequential except during recovery operations. It is also usually synchronous which means that transaction response time is highly dependent on fast access to the log. Thus, a pair of mirrored disks is probably the best solution for storing the log file while the database itself can be stored in a RAID5 or parity striping type array.

For each update access, the log records provide the disk name, the file name and the relative sector number within the file. The physical address of the first sector of a file is not available from the log. Therefore, a policy for allocating physical disk space to files is needed. In our experiments we allocate disk space using fixed size extents. The extent size used in our simulations is 1MB. At the beginning of each simulation run, the first extent allocated in each disk is chosen at random. When a new extent is needed, the next available extent is allocated. We assume that the disks are 50% full, and thus the next extent is considered available with probability 0.5.

We did not change the assignment of files to disks in the case of mirrored disks and parity striping. For parity striping it was assumed that enough space was available on the disks to store the parity. The number of disks in the array is assumed to be equal to the number of disks accessed in the trace. Usually an extra disk is needed so that space for the parity can be allocated on each disk. However, we did not add a disk because that would have required remapping the data over more disks and we wanted to preserve the original mapping obtained from the trace. The number of disks is five for all three traces. For the RAID5 organization the data of the five logical disks accessed in each trace are striped over

five physical disks in a straight forward fashion. It is also assumed in this case that there is enough space on the disks to store the files referenced in the trace and the parity information so no extra disks are used. This way both RAID5 and parity striping organizations have the same number of disks while the mirrored disk organization has twice as many disks. The striping unit used for the RAID5 system is one block.

The capacity of the buffer in terms of number of blocks is denoted by  $B$ . In our experiments  $B$  ranges from 50 to 5000 blocks. The block size used is 4KB.

## 4 Experiments

In a first stage we use the timestamps included in the log file to determine the arrival rates of the update accesses. The values of those timestamps are dependent on the characteristics of the system on which the data was collected and are influenced by the response time and bandwidth of its I/O subsystem. However, they still provide us with an approximation of the rate at which I/O is issued in a real application. But in this case, a throughput measure is not significant because there are idle periods in the trace which means that a slower I/O subsystem organization can catch up with a faster one during the idle periods. Hence, the execution time of the trace, and therefore the throughput are not good comparison measures.

To compare the three organizations, we use a different measure which reflects the cost of executing the I/O load of the application on each disk organization. This cost measure is the total disk utilization, i.e., the combined disk usage times for all the disks in a given organization. Table 3 contains the results of this experiment. The I/O cost is measured

in seconds and the average response time in milliseconds. The results are shown for three different values of the buffer size  $B = 50$ ,  $B = 500$ , and  $B = 5000$ . The first two rows of the table contain the results for the first trace (T1), the next two rows correspond to the second trace (T2) and the last two rows to the third trace (T3). Mirrored systems have much lower I/O cost than RAID5 and parity striping. This is mainly due to the fact that in the last two systems an extra rotation is needed to read the old data and old parity for each update access. The reason the I/O cost for parity striping is less than that of RAID5 is because the seek distances in the RAID5 case are longer. This is due to the fact that the logical disks are mapped to the RAID5 array in such a way that when a request for data from a given logical disk follows a request for a different logical disk a large number of cylinders have to be traversed, while in the case of parity striping logical disks and physical disks are the same and therefore consecutive requests to different logical disks do not necessarily cause large disk arm movements. I/O response times are also shorter for mirrored systems than for the two disk array organizations because of the extra rotation required in disk arrays to read the old data and old parity and also because the mirrored disk system has more disk arms (twice as many), hence less queuing delays than in the disk array systems. Response times for reads are also shorter for mirrored disks since the disk in the mirrored pair that has the shorter seek distance to the data is always used to perform the read. As far as comparing the response times of RAID5 and parity striping, there are two phenomena in effect. On one hand, the service time at each disk (seek + rotation + transfer) is longer for RAID5 because of the larger arm movements mentioned above. On the other hand there is less queuing in

Table 3: Results using timing information provided in the log.

		$B = 50$			$B = 500$			$B = 5000$		
		Mir	Raid	ParStr	Mir	Raid	ParStr	Mir	Raid	ParStr
T1	I/O cost	1902	4493	3913	1354	3635	2489	1058	2845	1931
	resp time	27.2	59.7	61.9	229.3	894.8	1055.8	262.2	957.6	1159.9
T2	I/O cost	6785	15289	13041	4212	9614	8230	1184	3376	2421
	resp time	20.1	43.4	38.9	35.3	157.9	160.9	506.3	4376.2	4265.6
T3	I/O cost	244	638	526	7.2	11.8	10.8	7.2	11.8	10.8
	resp time	19.0	48.3	40.6	21.6	49.3	48.6	21.6	49.3	48.6

RAID5 systems because of better load balancing and therefore better utilization of the disks in the array.

The inflated numbers for the average response time that appear in the table for  $B = 500$  and  $B = 5000$  for the first two traces are due to the long queuing delays encountered when checkpoints are generated. In our simulation, all the I/O requests performed by a given checkpoint are issued at the same time<sup>†</sup>. For a bigger database buffer, the checkpoint will generate more concurrent I/O and thus cause more queuing. Checkpointing I/O is done asynchronously and therefore long response times for this type of I/O is not a problem. However, normal transaction processing I/O (due to buffer replacement) is synchronous and has to meet stringent response time requirements. For larger  $B$ , the contribution of the checkpointing I/O to the average response time becomes much higher because there are more I/O's per checkpoint and their response time is longer while the amount of replacement I/O becomes smaller.

In a second experiment, we use a different approach to apply the traces to the disk

<sup>†</sup>The log file contains one record per checkpoint indicating the checkpoint has been completed.

organizations. We ignore the timestamps provided in the log. Instead we try to achieve a given average response time by controlling the number of I/O's allowed to enter the system. Then we measure the corresponding throughput and compare the throughput of different organizations for the same response time. Figure 3 shows the throughput vs. average response time curves for the three traces for three different buffer sizes. The main result is that under light load, parity striping performs better than RAID5 because of its shorter average seek times and the absence of any significant queuing delays. However, at higher loads, when queuing delays become more significant, RAID5 performs better mainly because of its load balancing advantage. The negative effects of striping in transaction processing environments outlined in [3], such as using many disk arms to service a single request and thus causing more queuing, do not materialize in this case because most requests are for single blocks and striping is done at the block level. The throughput of mirrored disks is much higher than that of the other two organizations because they have twice as many disk arms to service the requests and do not incur the overhead of the extra rotation for reading the old parity and old data. The "knee" in the curve occurs earlier for mirrors than for RAID5 because RAID5 systems are more load balanced and saturation occurs at a higher load.

As the size of the buffer  $B$  increases, the gap between the mirror curve and the RAID5 curve narrows while the gap between RAID5 and parity striping widens. This can be explained by the fact that when the working set of the application fits entirely in the buffer, most of the I/O is performed during changes in locality when a new more or less

contiguous area of one of the disks is brought into the buffer. Therefore, service time (seek+rotation+transfer) of the I/O becomes shorter while queuing time becomes a more important fraction of the total response time. RAID5 performance improves in this kind of an environment because of its better load balancing while Mirroring and parity striping do not do as well. The change in behavior occurs somewhere between  $B = 50$  and  $B = 500$  for the first trace and somewhere between  $B = 500$  and  $B = 5000$  for the second trace. The third trace is much shorter than the other two and does not exercise its database as much. The change in the shape of the curve for mirrors for  $B = 500$  and  $B = 5000$  for the last trace is due to the fact that the hit ratio in the buffer becomes very close to one (0.987 for  $B = 500$ ) which means that practically all the I/O is checkpointing I/O which exhibits different behavior than replacement I/O. The set of blocks that get modified between checkpoints is such that increases in throughput and better disk utilization cannot be obtained in the case of mirrors unless a large number of I/O are allowed into the system at the same time. RAID5, however, does not suffer from that problem because it stripes the data over all the disks and keeps them all busy.

The above evaluation does not account for the fact that mirrored disk systems use twice as many disks. To get a more fair comparison we look at the throughput per disk. This is done by dividing the throughput by the number of disks in the I/O subsystem organization. In Figure 4, the throughput per disk versus the average response time is plotted. We see that for large enough buffer size, RAID5 throughput per disk is higher than that of mirrored disks except at very light loads.

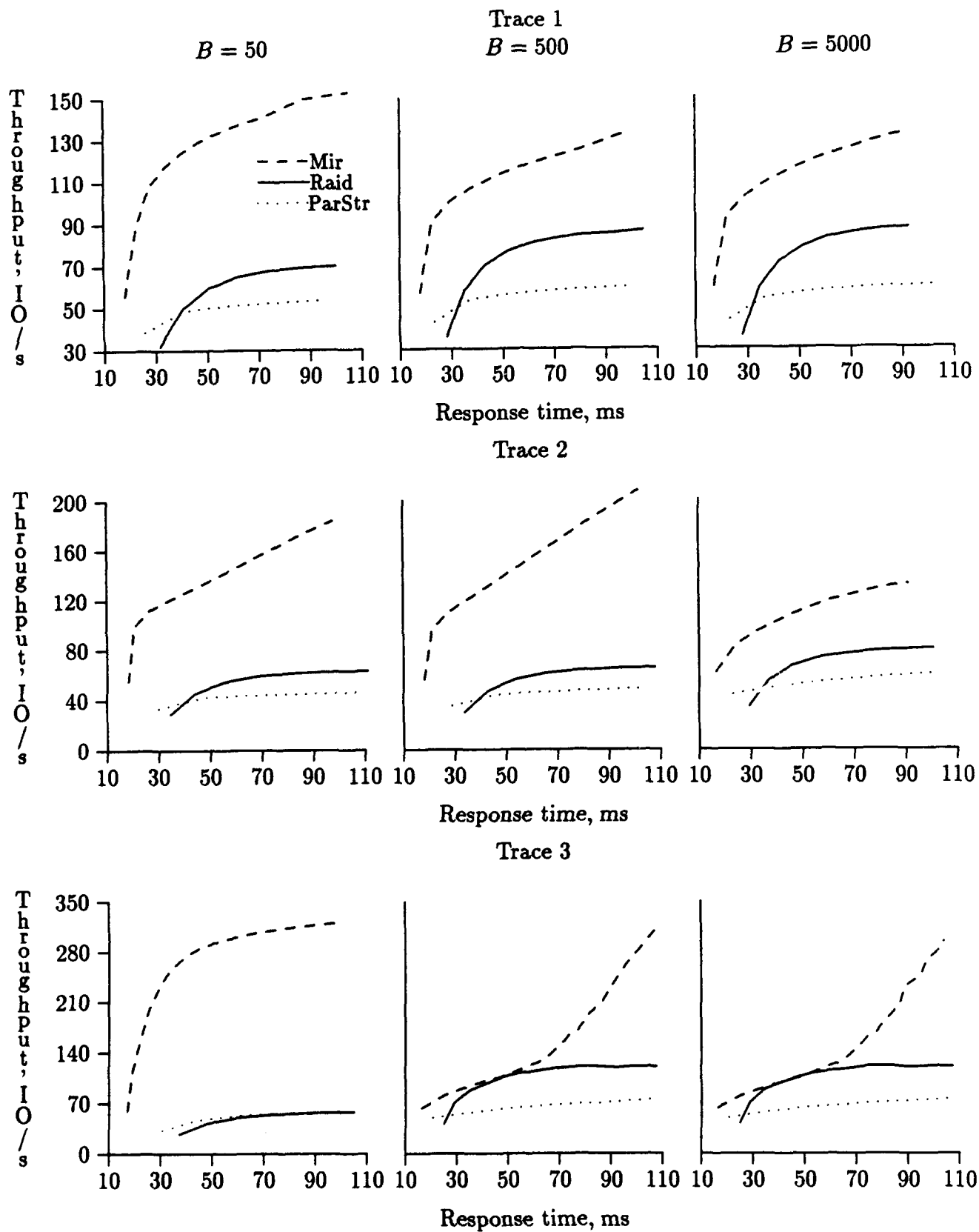


Figure 3: Throughput vs. response time.

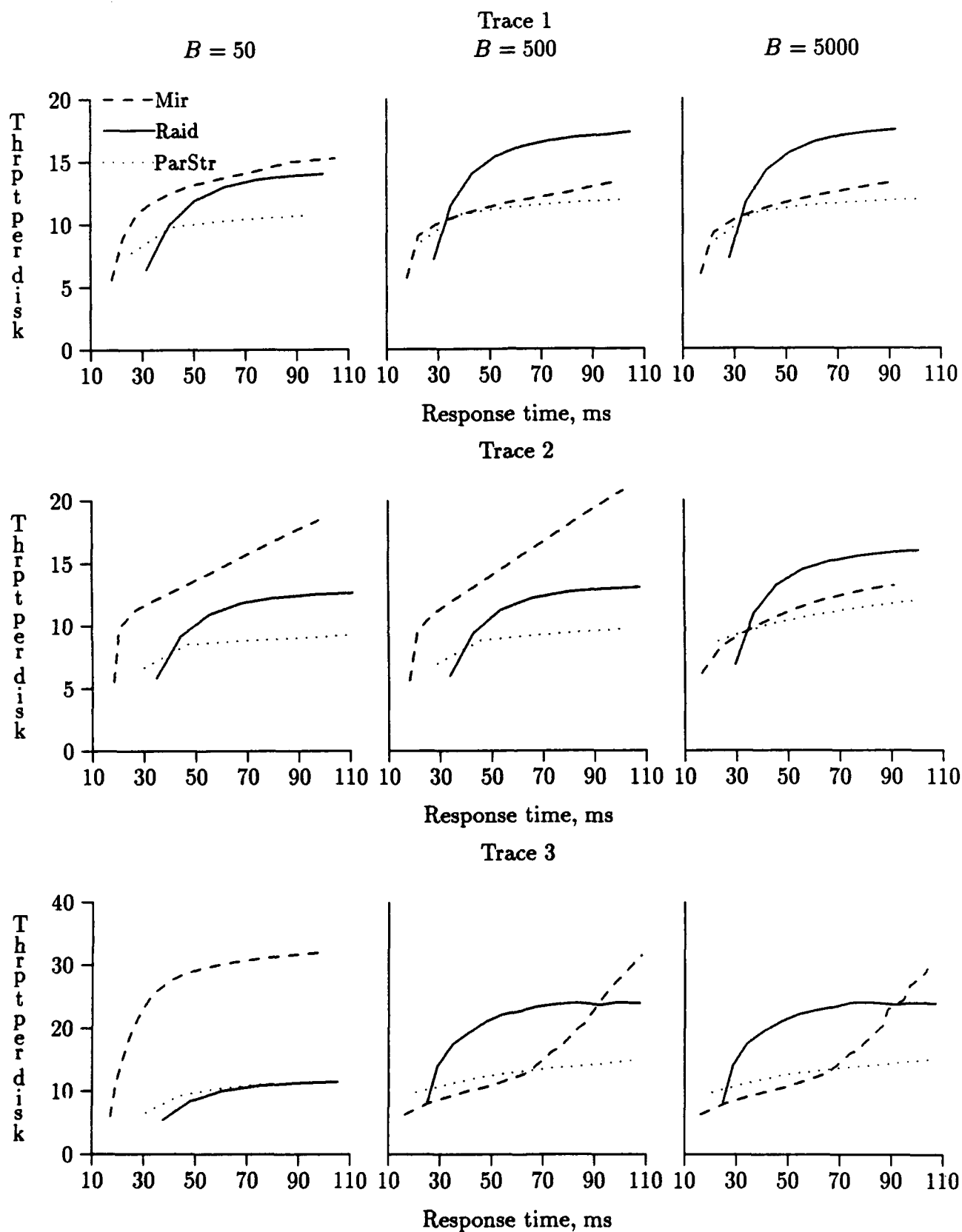


Figure 4: Throughput per disk vs. response time.

## 5 Conclusions

We have used data from operational transaction processing systems to evaluate the performance of two redundant disk array organizations and compare them to mirrored disks. We found that in an environment dominated by single block, I/O RAID5 provides generally better throughput than parity striping mainly because of its load balancing effect. Only under light load conditions does parity striping perform better than RAID5. Mirrored disks still provide significantly higher throughput but at a much higher storage cost. The gap in performance between mirrored disks and the disk array organizations narrows as buffer size increases.

## Acknowledgements

The authors would like to thank Robert Horst, Jim Carley, Srikanth Shoroff, Robert van der Linden, and Susan Whitford of Tandem Corporation for providing the TMF audit tapes and for answering numerous questions.

## References

- [1] BITTON, D., AND GRAY, J. Disk shadowing. In *Proceedings of the 14th International Conference on Very Large Data Bases* (Sept. 1988), pp. 331-338.
- [2] CHEN, P. M., GIBSON, G. A., KATZ, R. H., AND PATTERSON, D. A. An evaluation of redundant arrays of disks using an Amdahl 5890. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (May 1990), pp. 74-85.
- [3] GRAY, J., HORST, B., AND WALKER, M. Parity striping of disk arrays: Low-cost reliable storage with acceptable throughput. In *Proceedings of the 16th International Conference on Very Large Data Bases* (Aug. 1990), pp. 148-161.

- [4] HAERDER, T., AND REUTER, A. Principles of transaction-oriented database recovery. *ACM Computing Surveys* 15, 4 (Dec. 1983), 287-317.
- [5] KIM, M. Y. Synchronized disk interleaving. *IEEE Trans. Comput.* C-35, 11 (Nov. 1986), 978-988.
- [6] LIVNY, M., KHOSHAFIAN, S., AND BORAL, H. Multi-disk management algorithms. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems* (May 1987), pp. 69-77.
- [7] MENON, J. M., AND MATTSON, R. L. Performance of disk arrays in transaction processing environments. In *Proceedings of the 12th International Conference on Distributed Computing Systems* (June 1992).
- [8] PATTERSON, D., GIBSON, G., AND KATZ, R. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the ACM SIGMOD Conference* (June 1988), pp. 109-116.
- [9] REDDY, A., AND BANERJEE, P. An evaluation of multiple-disk I/O systems. *IEEE Trans. Comput.* 38, 12 (Dec. 1989), 1680-1690.
- [10] SALEM, K., AND GARCIA-MOLINA, H. Disk striping. In *Proceedings of the IEEE International Conference on Data Engineering* (Feb. 1986), pp. 336-342.
- [11] STONEBRAKER, M., KATZ, R., PATTERSON, D., AND OUSTERHOUT, J. The design of XPRS. In *Proceedings of the 14th International Conference on Very Large Data Bases* (Sept. 1988), pp. 318-330.
- [12] TANDEM COMPUTERS INC. *Introduction to the Transaction Monitoring Facility (TMF)*. Cupertino, CA. Part Number 15996.